به نام خدا

پروژه برنامه‌نویسی فاز اول درس «اصول طراحی کامپایلر»

نیمسال اول سال تحصیلی ۱۴۰۳-۱۴۰۲

دانشکده مهندسی و علوم کامپیوتر تاریخ تحویل: ۱۴۰۲/۰۹/۰۶ ساعت ۲۳:۵۹

در این پروژه قصد داریم، یک‌سری دستورات ساده و مرکب[1] برای یک زبان برنامه‌نویسی فرضی طراحی کنیم که در آن چند مرحله زیر به ترتیب در کلاس‌های مختلف با استفاده از زیرساخت LLVM قابل اجرا بوده و در نهایت object code به وسیله زیرساخت LLVM قابل تولید و نمایش باشد. زبان برنامه‌نویسی فرضی ما دارای دو نوع از دستورات ساده و مرکب است.

- دستورات ساده
  - تعریف متغیر
  - انتساب متغیر
  - عملیات ریاضی
- دستورات مرکب
  - شرط
  - حلقه

الف) دستورات ساده

تعریف و انتساب متغیر

در زبان طراحی شده، متغیرها دارای مقادیری هستند که در زمان compile مشخص خواهد شد. برای تعریف متغیر، نکات زیر را رعایت کنید:

- تنها data type موجود در این زبان‌برنامه‌نویسی integer بوده و مقدار پیش‌فرض آن 0 است.
- متغیر باید با کلیدواژه int تعریف شود. با استفاده از , می‌توان چند متغیر را هم‌زمان تعریف کرد و در انتها حتما باید از ; استفاده شود.

$$int\ a_1,\ ...,\ a_n\ =\ e_1,\ ...,\ e_m;\ //\ (n \geq m)$$

- قبل از انتساب، متغیر حتما باید تعریف شده باشد.
- هنگام تعریف متغیر می‌توان چند متغیر را هم‌زمان با در نظر گرفتن قوانین زیر مقداردهی کرد:
  - تعداد مقادیر، برابر تعداد متغیرها باشد:

[1] Simple and compound statements

$int\ a,\ b\ =\ 1,\ 2;\ \ //\ a\ =\ 1,\ b\ =\ 2$

- تعداد مقادیر، کوچک‌تر از تعداد متغیرها باشد:

$int\ a;\ //\ a\ =\ 0$
$int\ a,\ b,\ c;\ //\ a\ =\ 0,\ b\ =\ 0,\ c\ =\ 0$
$int\ a,\ b,\ c\ =\ 1;\ //\ a\ =\ 1,\ b\ =\ 0,\ c\ =\ 0$
$int\ a,\ b,\ c\ =\ 1,\ 2;\ \ //\ a\ =\ 1,\ b\ =\ 2,\ c\ =\ 0$

- تعداد مقادیر، بزرگ‌تر از تعداد متغیرها باشد:

$int\ a,\ b\ =\ 1,\ 2,\ 3;\ \ //\ Syntax\ error$

- می‌توان متغیر را با هر عبارت ریاضی که در زبان، معتبر باشد مقداردهی کرد:

$int\ a;$
$int\ b\ =\ a\ *\ 2\ +\ 6;$

- می‌توان عملیات انتساب را با عملگرهای محاسباتی ترکیب کنید:

$int\ a\ \ +=\ \ 2\ \ +\ \ (3\ \%\ x);$
$int\ a\ \ *=\ \ x\ \hat{}\ (3\ +\ a\ \%\ 4);$
$int\ a\ \%\ =\ \ 7\ \%\ d;$

عملیات ریاضی

عملیات ریاضی عبارتی است که در نهایت به یک مقدار ارزیابی می‌شود. این عبارت شامل موارد زیر خواهد بود:

- پرانتز ( )
- توان ^
- عملگرهای محاسباتی
  - ضرب، تقسیم، باقیمانده $*,\ /,\ \%$
  - جمع، تفریق $+,\ -$
- عملگرهای رابطه‌ای
  - بزرگتر، کوچکتر $>,\ <$
  - بزرگتر مساوی، کوچکتر مساوی $>=,\ <=$
  - مساوی، نامساوی $==,\ !=$
- عملگرهای منطقی
  - عطف and
  - فصل or
- عملگرهای انتساب $=,\ +=,\ *=,\ /=,\ -=$

عملگرها به ترتیب اولویت نوشته شده‌اند پس توجه داشته باشید که اولویت عملگرها را بطور صحیح رعایت کنید.

ب) دستورات مرکب

شرط

شرط‌ها ساختاری مرکب دارند که جریان برنامه را مدیریت می‌کنند. ساختار کلی یک شرط به صورت زیر خواهد بود:

- بلوک شرط با کلیدواژه if آغاز می‌شود سپس شرط مورد نظر نوشته شده و در آخر از : برای پایان شرط استفاده می‌شود.

$if\ condition$:

- بدنه شرط با استفاده از $begin$ آغاز و با end پایان می‌یابد.
- بطور مشابه از $elif\ condition$: و $else$: استفاده می‌کنیم.

$if\ c_1$: $begin$
...
end
$elif\ c_2$: $begin$
...
end
$else$: $begin$
...
end

- توجه کنید که بلوک‌ها با $begin$ و end مشخص می‌شوند و دندانه‌گذاری و $\backslash n$ تاثیری ندارند.

$if\ c1$: $begin$ ... end ✔

- $c_{1,...,n}$ یک عبارت ریاضی هستند و به یک مقدار ارزیابی می‌شوند. در صورتی که این مقدار برابر صفر باشد شرط برقرار نبوده و در غیر این صورت برقرار خواهد بود.
- به دلیل سهولت کار در بدنه شرط، مجاز به تعریف متغیر نیستیم. مثال زیر را در نظر بگیرید:

$int\ a\ =\ 2;$
$int\ b\ =\ 3;$

```
if a + b > 6: begin
        a = 1;
end
elif a + b < 0: begin
        b = 1;
        a = 0;
end
else: begin
        b = 2;
end
```

حلقه

حلقه‌ها دارای ساختاری مرکب هستند که از تکرار جلوگیری می‌کنند. ساختار کلی حلقه به صورت زیر تعریف می‌گردد:

- بلوک حلقه با کلیدواژه $loopc$ آغاز می‌شود. سپس یک عبارت ریاضی نوشته شده و در آخر از : برای پایان حلقه استفاده می‌شود.

```
loopc c_1:
```

- بدنه حلقه با استفاده از $begin$ آغاز و با end پایان می‌یابد.

```
loopc c_1:
begin
...
end
```

- توجه کنید که بلوک‌ها با $begin$ و end مشخص می‌شوند و دندانه‌گذاری و $\n$ تاثیری ندارند.

```
loopc c_1: begin ... end ✔
```

- c_1 یک عبارت ریاضی است و به یک مقدار ارزیابی می‌شود. در صورتی که این مقدار برابر صفر باشد شرط حلقه برقرار نبوده و در غیر این صورت برقرار خواهد بود.
- به دلیل سهولت کار در بدنه حلقه، مجاز به تعریف متغیر نیستیم. مثال زیر را در نظر بگیرید:

```
int x;
loopc x * 12 + (2 ^ x) < 2 ^ 8:
begin
x += 12
```

end

نکات مهم:

- بهتر است از زبان C یا C++ برای توسعه کدها استفاده کنید.
- حتما در پیاده‌سازی‌ها، خطاهای استاندارد در بخش‌های مختلف را پیش‌بینی کنید.
- توسعه هر بخش در فایل‌های جداگانه انجام شود که قابلیت ارزیابی مجزا را داشته باشد.
- کدها خوانایی مناسبی داشته باشد و پیشنهاد می‌شود به درستی کامنت‌گذاری شود.
- کدهای خروجی به همراه سند تشریحی و نمونه خروجی‌های اجرا شده در یک ریپوی GitHub با دسترسی خصوصی، قرار داده شده و لینک آن به عنوان خروجی، ارسال شود. همچنین لازم است حساب کاربری alidoostnia به عنوان همکار به ریپوی Github افزوده شود.
- پروژه به صورت تیمی قابل اجرا است. اندازه تیم‌ها بین ۲ الی ۳ نفر قابل قبول است و در فاز بعدی، امکان تغییر اعضای تیم وجود ندارد.
- همه اعضای تیم باید در انجام پروژه مشارکت داشته باشند و تسلط هر فرد جداگانه ارزیابی خواهد شد.
- جهت پیاده‌سازی درست و کامل پروژه، پیشنهاد می‌شود تا اسناد مرتبط با سایت LLVM با دقت مطالعه شده و همچنین سه فصل اول کتاب Learn LLVM 12 مطالعه شود. کدهای نمونه و روش ارائه مطالب در کتاب به درک شما از زیرساخت یک کامپایلر مینیمال کمک خواهد کرد.

«موفق باشید»